CREAM ULDB Mission: Computing, Data Formats, & Communication Draft ICD

M. A. DuVernois
School of Physics and Astronomy
University of Minnesota
Minneapolis, MN 55455

1. Executive Summary

The flight and ground support computer systems and the data acquisition setup are under construction at the University of Minnesota. The School of Physics and Astronomy machine shops are building the flight housing for the system, software is under development in house, and the electronics systems are working on the bench. Interfaces to other systems are largely developed, with a few gaps that are being resolved in real-time.

1.1 Flight computer

The flight computing system consists of an Adlogic (Advanced Digital Logic) MSEBX-P5-266-ESACF Intel Mobile Pentium based PC/104+ bus computer. It is connected via a PC/104+ stack to modules responsible for communications with instrument and detector systems. Data will flow in through fast serial (RS-485 protocol) connections from the TCD. TRD readout will likely be handled through a PC/104 DSP (TI TMS320-based system from Signal Logics Corp.) board with dual-port RAM. Data from the calorimeter will arrive in a pair of (to-be-constructed) custom (dual-port memory) PC/104 board. Instrument control will be carried out with digital I/O modules as well as via messages passed from NASA via the Ethernet link. Housekeeping data and detector functional commanding are transferred over a pair of RS-232 interfaces between Maryland's housekeeping processor and the main flight computer. The data will leave the flight computer system over Ethernet to the balloncraft systems for transmission. The operating system is a TimeSys Linux-RT built up on a hard real-time microkernel. Most components will be available commercially as off-the-shelf parts.

1.2 Ground support system

The ground support computer system is being built with the paired purpose of supporting and debugging the flight computer as well as serving as a ground-based substitute for the flight computer system. The system architecture will be PCI-based industrial computing built around a Cyber Research Pentium-III PICMG single board computer, National Instruments and Computer Boards acquisition boards, LabView instrument control software, and a dual-network (outside world & instrument) topology. This computer system can be used in three different modes: analyzing the end-result data coming out of the flight system over Ethernet, monitoring data flow over Ethernet after it leaves NASA ground support (flight operation mode), and completely substituting for the PC/104 stack and carrying out all of the acquisition duties directly in PCI. The whole ground support system will be ruggedized and racked-mounted for easy transport and quick setup in the field.

2. Overall system design

The basic principals of the design of the system are as follows:

1. COTS, commercial off-the-shelf hardware
2. Maximum speed and processing power available within the slow (PC/104) bus
3. Simple and reliable software tools
4. Reasonable cost
5. Flexible data acquisition
6. Hardware that we're pretty familiar with

2.1 Hardware

2.1.1 Computers

The flight hardware was constrained to the rather outdated PC/104 standard. Within that framework, we chose to proceed with the maximum computer speed and performance without harming the power budget. The main CPU uses laptop components to keep power consumption under 10W at an operating speed of 266MHz internal, 133MHz to cache, 66MHz to memory, and 33MHz to bus, all with 32 bit words. The computer is in the EBX hardware format—somewhat larger than the traditional PC/104 footprint but fully supporting the PC/104 (and PC/104+) data standards. The only other available 266MHz Pentium processors in PC/104 use desktop components and have significantly higher power consumption. (Since this design decision was made, Pentium-II and III laptop component PC/104 boards have become available. Their additional processing power has not been needed however.)

The ground support computer was chosen as a reasonable, available single board computer from a supplier that we've previously dealt with (Cyber Research). This system should be sufficiently powerful to pull in all of the experiment data, display summaries and diagnostics, and perform routine analysis work simultaneously. The multiple modes that we've required the system to perform allow the flight computer and the flight acquisition systems to be tested *in situ* without significant changes made to the payload. This also allows testing of single detector system or the entire experiment without the flight computer in place, or with different DAQ software. The ground support station also supports PC/104 and PC/104+ boards either stacked with the CPU board or in PC/104 to ISA adapter board or PC/104+ to PCI adapter board format.

2.1.2 I/O

Flight DAQ I/O will come in over the PC/104 and PC/104+ buses with the exception of a few ports on the flight computer itself (RS-232 and Ethernet). Additional I/O boards are as follows: RS-485 for reading out the TCD (four ports), Globalbus wide-fast-digital input to the dual-port memory of the DSP from the TRD, input to the custom dual-port memory units for the calorimeter, RS-232 to the housekeeping and commanding system, digital I/O for resets, trigger conditions, event clears, GPS antenna input, and normal computer ports for on-the-ground diagnosis and connected operating mode.

2.1.2.1 Serial/Parallel Digital

Fast serial communications with the PC/104 bus are possible with a number of interface cards available from a multitude of vendors. We are using Emerald series cards from Diamond Systems for the readout. The TCD uses RS-485 serial communications protocol in the multiple-drop configuration (party-line extension to RS-422). Each segment of the timing-charge detector can communicate over the same communications bus and will be configured to minimize data collisions and maximize throughput. The TRD is to be readout through a PC/104 coprocessor—a 40MHz TI DSP with dual port RAM and 32 bit no-wait state parallel data highway. It is then read onto the main processor with maximum speed transfers.[1]

2.1.2.2 Discrete Digital

Triggering will take place over digital I/O lines that can be asserted on either the computer or detector side. Nominally we plan on having a total of 48 such lines implemented on a single PC/104 board (which will also host the livetime scalars and timers). Additional digital lines can be made available if there is an interest/need. The trigger bits will include information on which detector systems the flight computer should expect to find data from.

2.1.2.3 Analog

There will be a minimum of 16 scanning ADC channels (read out slowly, e.g., as housekeeping events, once every few seconds) and 8 channels of DAC (set at the same rate). Additional channels can also be easily added to the system for slow control loops, setting high voltages, etc. as needed by detector systems and housekeeping systems although the main housekeeping flow will be through the Maryland housekeeping processor unit. These SCADA lines will be read and set by the ground support station running LabView in this mode. The CPU motherboard also has three analog video (NTSC/PAL) inputs as well. It's not clear if there's a good use for these, but they are available and can write single video frames at a low rate without impacting livetime.

2.1.2.4 Custom

The calorimeter is planning on bringing the data directly to the PC/104 bus using dual-port memory. A second copy of these boards would be useful to have for the ground support station.

2.1.2.5 GPS

The Adlogic CPU has a built-in GPS receiver (built on a Rockwell PC1-2-GPS design and chipset). The controller for this GPS module is flashable and Rockwell can provide us with a version of its O/S which will allow for altitudes > 70kfeet or speeds in excess of

[1] Other detector systems and subsystems are encouraged to communicate with the flight computer and DAQ over commercial serial buses. We can support short message passing over RS-232, data collection over RS-422 and RS-485, fast parallel digital data readout, serial protocols over Ethernet, and burst transmissions (with a token-key system) over USB. PC/104 board solutions also exist for CANbus, SCSI, ARINC-429/629, STANAG 3910, IEEE-1394, SSA, RS-449/530, bluetooth, X.21, high-speed parallel, and 4-20 mA current loop. These boards are COTS and available, typically, for a few hundred dollars.

250MPH. This will allow us to integrate GPS time into our main data stream rather than as a post-flight data analysis data merge.

2.1.3 Network

There will be three networks between the flight science computer, ground support computer, Wallops flight computer side of the payload, and the outside world. These will be the Ethernet between the flight computer, ground support, and NSBF computing. Data on this network will either be in ready-to-be-processed-by-balloocraft-for-transmission format. Data will be generated at the flight computer end and transmitted to two receivers, and flight commanding will be generated at the NSBF communications computer and received by the flight and ground support computers. While on the ground this network allows for software updates, data processing off the payload, and a direct monitor of PC/104 bus activity. The final network is the ground support computer's connection to the outside world. This connection will allow for monitoring of the payload at collaborator institutions but will not allow for direct modifications of the flight software (for security/hacker reasons). The single board computer has two Ethernet ports and dual controllers.

2.1.4 Other ground support

A ground support power supply will be required to simulate the battery power output. This will be controllable by the ground support station over GPIB, or manually, depending on the test mode.

2.1.5 Flight DC/DC converters

A high-efficiency PC/104 DC-DC power converter optimized for mobile 24-28VDC applications at extended commercial temperatures from Diamond Systems (in coordination with MIL-SPEC folks at Tri-M) is used for flight.

2.2 Software

2.2.1 O/S

Real-time Linux has the advantages of low-cost, robust construction, ease of use of software tools, availability of device drivers, and familiarity to the collaboration. We have purchased a commercial microkernel for Linux-RT from TimeSys to ensure that those advantages are coupled to a hard, deterministic real-time tasking kernel. The microkernel is from VXWorks that has a well-known hard-real-time reputation (not to mention actual performance). This has traded the low-cost (free now up to a couple hundred $) for a deterministic system.

2.2.2 Flight code

The flight codes are optimized for the flight computer through a self-hosted development process. Full timing simulations will also be performed once data simulators are provided from each detector group (see 2.2.3). Flight software and O/S will be flashed onto disk-on-chip (DOC) devices. Software revision control will be established at Minnesota for the flight software. The provision for re-flashing the DOC during flight will tentatively be enabled with a fail-safe mode to prevent erasure. (Basically, the new flash code is set into RAM, check-summed, tested through one operation cycle, and tested for "undo-ability"

before truly being flashed onto the DOC.) Two DOCs are supported as multiple boot sources. Installed on the computer we have 128MB of SDRAM, 128MB of Flash Disk (used for short-term nonvolatile storage of event data and tables), and two 144MB DOCs.

2.2.3 Development tools

The principal software development tools for the real-time code to run on the Linux-RT flight computer are the cross-compilers for gcc, the TimeSys and Embeddix timing simulators, and the embedded driver library for Linux-RT. Development will be largely self-hosted using a ground-only PC/104 stack, power supplies, and interconnects. Timing simulations will run on the ground support computer and the Minnesota Particle Astrophysics Beowulf cluster.

2.2.4 LabView

Ground test data from the instruments and housekeeping will be acquired in a simple LabView environment for ease of hardware and hardware-software interface debugging. This environment can be maintained during flight for monitoring the instrument housekeeping. LabView drivers exist for all of the PCI-based acquisition modules used in the ground station.

2.2.5 Analysis tools

Initial data formatting will be implemented in either CERN Root (using C++) or CERNLib (PAW and Fortran code) depending on whom exactly does the coding. Much of the existing CERNLib code can be used in a relatively straightforward way under Root. The Root learning curve isn't huge. It is anticipated that the instrument Monte Carlo will be set up in a compatible manner, either GEANT 3.2.1 or GEANT 4 for PAW and Root respectively.

2.2.6 Event display

The event display software is in the early stages of development, but will be implemented as a VB GUI which monitors incoming data at the TCP port level and displays current housekeeping, periodic events, and histograms housekeeping and event parameters for examination of time-evolution.

3 Interface definition

The physical and logical (data) layers for the connections between the flight science computer and the rest of the balloon detector systems and balloonicraft systems are not fully defined yet. Most of the remaining issues however, are expected to be resolved in the next couple of months as systems are integrated together.

3.1 Physical layer

3.1.1 Science to/from flight computer

An Ethernet interface (10 or 100 Mbit available on science computer end) connects the flight system computer to the balloonicraft systems. On that end, there are multiple computers and an Ethernet hub/switch. The same connection, or that connection routed through an external switch, will be employed in the field for ground support.

3.1.2 Housekeeping/commanding

The Maryland housekeeping and commanding systems interface to the flight science computer using two RS-232 connections with DB9 connectors on the computer box. Power will be supplied over those connections as well to run the housekeeping and commanding units.

3.1.3 TCD

Multiple RS-485 interfaces (up to 4) each with multiple drops (at individual PMTs) will carry the PHA (slow) data from the timing charge detector. All physical port connectors will be DB9 on the flight computer housing.

3.1.4 TRD

The fast-wide digital bus between the TRD readout system and the DSP's dual-port memory will be cabled through a TBD connector at a TBD data-rate. (DSP and memory is chosen and the readout system from Chicago will provide a suitable input to it.)

3.1.5 Calorimeter

The calorimeter readout modules in PC/104 are custom units that will concentrate the calorimeter data into two non-interrupting PC/104 dual-port memories. These memories will be read across the 16-bit side of the ISA-like PC/104 interface. Cable connections through the computer case are TBD.

3.1.6 Trigger

Discrete digital lines will be set for a trigger. Distinct lines will indicate which portions of the detector system should have data. The flight science computer will clear those lines when it is ready for a new event.

3.1.7 Environment

A GPS receiver is built onto the science flight computer. It is flashable with a secured microkernel that would allow for acquisition at altitudes exceeding 70kfeet. A couple of thermal sensors tightly integrated to the computer system will be read out to monitor the computer performance and immediate environment. Watchdog timers will ensure a smooth restart to the acquisition should a software priority inversion occur.

3.2 Data layer

3.2.1 Science to/from flight computer

(Section by Chris Lewis)
Data passes between the science and flight computers by means of an Ethernet connection. A stand-alone network is realized with the use of PC104 combination network interface and multi-port repeaters. These NIC/hubs reside on the PC104 bus of the flight computers.

Universal Datagram Protocol or UDP is used to minimize difficulty maintaining a connection between science and flight. In the event of loss of power to one of the computers, reconnection is not necessary.

Unlike Transmission Control Protocol, UDP does not perform any handshaking. An application-level handshaking protocol has been put in place so as not to lose any data. After a packet is sent, a reply indicating the successful receipt of that packet is sent by the receiving computer. Information about the format of these packets will be given later in this document.[2]

Let's assume the flight computer was to send a packet to the science computer and the science computer was unable to process it, the flight computer is expected to send the same packet again. The flight computer will resend the packet every seconds until either confirmation is received or thirty attempts have been made. If the packet is not confirmed after thirty seconds the flight computer will ping the science computer to ascertain its status. Science is expected to resend packets in a similar fashion.

If during this packet confirmation process new data has become available, it must be held in a queue or similar data structure until older data can be processed.

There are essentially five different kinds of packets in two different groups. One group is numbered and has a cyclic redundancy check performed on it. This group is populated by data, command, and housekeeping packets. The other group is made up of connection status packets and packet confirmation packets.

Data, command and housekeeping packets are characterized by a packet payload header. This header is eight bytes in length. The first two bytes are used to identify the packet type.

Data:	0x05, 0xFA
Housekeeping:	0xAF, 0x50
Command:	0x50, 0xAF

The second two bytes are the result of the cyclic redundancy check or CRC. These two bytes are written as a sixteen-bit little endian number. In other words, the least significant byte is first; the most significant is byte second. Tables as well as the function used to generate the CRC are contained in this document.

The remaining four bytes are used as a counter. This number is also written in little endian. Every time a packet in this group is sent, the counter is incremented.

[2] This should allow UDP to be nearly as reliable as TCP, though with significantly higher overhead.

Whenever data is sent to a flight computer by a science computer, this header is stripped off before the data is made available. Use of the Wallops socket library will strip this header off automatically.[3]

After a packet from this group is sent, the computer receiving it sends a packet confirmation. These packet confirmation packets are used to ensure that no data is lost. The payload on these packets is just four bytes in length. They have a two-byte packet type field, or sync pattern just as the previous packets.

Packet Confirmation: 0xFA, 0x05

This field is followed by the least significant two bytes of the four-byte counter from the packet being confirmed. In other words if packet number 0x20C733D1 is received, the confirmation packet will look like this: 0xFA, 0x05, 0xD1, 0x33.

The remaining packet is the connection status packet. These packets are used to establish the well being of the computer you are connected to. They are simply the two-byte sync pattern used to differentiate the packet types.

Connection Status: 0xA0, 0x5F

Status packets are sent by each computer at a rate of one every five seconds. If after thirty seconds the flight computer has not received a connection status packet, it will ping the science computer to find out if it has lost power.

Connection status packets serve another important function. In the universal datagram protocol the server does not know the port number on the client until it receives data from it. The science computer, as the server, will normally need to send data before receiving a command. So this mechanism helps to activate the connection between server and client.

The Consultative Committee for Space Data Systems or CCSDS has established a protocol used by the Ultra Long Duration Balloon. It is important to note, at present, a packet cannot cross a CCSDS frame boundary. Our implementation of this protocol will require you to limit the size of your packets to approximately 450 bytes.

3.2.2 Housekeeping/commanding

Standard RS-232 plus power lines. No handshaking, no dataflow control.

3.2.3 TCD

Standard RS-485 (extension of RS-422). No additional power loads.

[3] Within the data block, the data will have already been processed through a bit-error detector and correct algorithm optimized for multiple-bit running errors associated with radio communications. This function is transparent to both the data analyzer (this packetization is stripped off when the data is written on the ground) and to the flight computer systems which are only looking for a stream of binary data. They never examine the data contents.

3.2.3.1 TCD Data Format
The data format for the TCD

3.2.4 TRD
32-bit wide data path. Data format unknown.

3.2.4.1 TRD Data Format
The data

3.2.5 Calorimeter
Data format in appendix. Fully formed event will be read from the dual-port memories.

3.2.6 Trigger
Digital signals will be recorded with timing separate from the data timing.

3.2.7 Environment
This data will be incorporated into the main housekeeping data blocks and, if appropriate, the error log, queue, and notification service.

3.3 Combined event data format
The combined data format

4 Vital stats for flight science computer system

Power:	<26W of 28VDC nominal (14-36VDC efficient, 8-40VDC tolerable) Computer: 8W Fast acquisition board: 5W each (2 needed) Slow acquisition boards: 1W each (4 needed) DC/DC converter efficiency: 2W
Weight:	10lbs (mostly in heat sinks/cabinet)
Size:	8”x8”x12” (plus cable clearances)
Thermal:	passive, conductive, heat passed to base plate
EMI/EMC:	Absolute measurement required. Need to find no evidence of 2, 8, 33, 66, 133, or 266 MHz noise at detector outputs.

5 Flight error reporting
Errors detected at the science flight computer level are prioritized as notifications, bound mismatches (parameters creeping outside of established limits), serious, or fatal. All non-fatal messages may be masked from the ground by command. This will introduce summarized error logging for that (or those) parameter(s)—“Computer too hot, error repeated 1534 times over the last hour,” for example. Non-repeated error messages have higher priority than data, housekeeping blocks are in a stack for downlink, and events are in a FIFO for downlink.

Appendix A:
CalcCRC (balloonicraft to/from science computer)

```
/* Begin CalcCRC.hpp ======================================== */

static const unsigned char lowCRC_ab[] =
{
        0x00, 0x21, 0x42, 0x63, 0x84, 0xa5, 0xc6, 0xe7, 0x08, 0x29, 0x4a, 0x6b,
0x8c,
        0xad, 0xce, 0xef, 0x31, 0x10, 0x73, 0x52, 0xb5, 0x94, 0xf7, 0xd6, 0x39,
0x18,
        0x7b, 0x5a, 0xbd, 0x9c, 0xff, 0xde, 0x62, 0x43, 0x20, 0x01, 0xe6, 0xc7,
0xa4,
        0x85, 0x6a, 0x4b, 0x28, 0x09, 0xee, 0xcf, 0xac, 0x8d, 0x53, 0x72, 0x11,
0x30,
        0xd7, 0xf6, 0x95, 0xb4, 0x5b, 0x7a, 0x19, 0x38, 0xdf, 0xfe, 0x9d, 0xbc,
0xc4,
        0xe5, 0x86, 0xa7, 0x40, 0x61, 0x02, 0x23, 0xcc, 0xed, 0x8e, 0xaf, 0x48,
0x69,
        0x0a, 0x2b, 0xf5, 0xd4, 0xb7, 0x96, 0x71, 0x50, 0x33, 0x12, 0xfd, 0xdc,
0xbf,
        0x9e, 0x79, 0x58, 0x3b, 0x1a, 0xa6, 0x87, 0xe4, 0xc5, 0x22, 0x03, 0x60,
0x41,
        0xae, 0x8f, 0xec, 0xcd, 0x2a, 0x0b, 0x68, 0x49, 0x97, 0xb6, 0xd5, 0xf4,
0x13,
        0x32, 0x51, 0x70, 0x9f, 0xbe, 0xdd, 0xfc, 0x1b, 0x3a, 0x59, 0x78, 0x88,
0xa9,
        0xca, 0xeb, 0x0c, 0x2d, 0x4e, 0x6f, 0x80, 0xa1, 0xc2, 0xe3, 0x04, 0x25,
0x46,
        0x67, 0xb9, 0x98, 0xfb, 0xda, 0x3d, 0x1c, 0x7f, 0x5e, 0xb1, 0x90, 0xf3,
0xd2,
        0x35, 0x14, 0x77, 0x56, 0xea, 0xcb, 0xa8, 0x89, 0x6e, 0x4f, 0x2c, 0x0d,
0xe2,
        0xc3, 0xa0, 0x81, 0x66, 0x47, 0x24, 0x05, 0xdb, 0xfa, 0x99, 0xb8, 0x5f,
0x7e,
        0x1d, 0x3c, 0xd3, 0xf2, 0x91, 0xb0, 0x57, 0x76, 0x15, 0x34, 0x4c, 0x6d,
0x0e,
        0x2f, 0xc8, 0xe9, 0x8a, 0xab, 0x44, 0x65, 0x06, 0x27, 0xc0, 0xe1, 0x82,
0xa3,
        0x7d, 0x5c, 0x3f, 0x1e, 0xf9, 0xd8, 0xbb, 0x9a, 0x75, 0x54, 0x37, 0x16,
0xf1,
        0xd0, 0xb3, 0x92, 0x2e, 0x0f, 0x6c, 0x4d, 0xaa, 0x8b, 0xe8, 0xc9, 0x26,
0x07,
        0x64, 0x45, 0xa2, 0x83, 0xe0, 0xc1, 0x1f, 0x3e, 0x5d, 0x7c, 0x9b, 0xba,
0xd9,
        0xf8, 0x17, 0x36, 0x55, 0x74, 0x93, 0xb2, 0xd1, 0xf0
};

static const unsigned char highCRC_ab[] =
{
        0x00, 0x10, 0x20, 0x30, 0x40, 0x50, 0x60, 0x70, 0x81, 0x91, 0xa1, 0xb1,
0xc1,
        0xd1, 0xe1, 0xf1, 0x12, 0x02, 0x32, 0x22, 0x52, 0x42, 0x72, 0x62, 0x93,
0x83,
        0xb3, 0xa3, 0xd3, 0xc3, 0xf3, 0xe3, 0x24, 0x34, 0x04, 0x14, 0x64, 0x74,
0x44,
        0x54, 0xa5, 0xb5, 0x85, 0x95, 0xe5, 0xf5, 0xc5, 0xd5, 0x36, 0x26, 0x16,
0x06,
        0x76, 0x66, 0x56, 0x46, 0xb7, 0xa7, 0x97, 0x87, 0xf7, 0xe7, 0xd7, 0xc7,
0x48,
        0x58, 0x68, 0x78, 0x08, 0x18, 0x28, 0x38, 0xc9, 0xd9, 0xe9, 0xf9, 0x89,
0x99,
        0xa9, 0xb9, 0x5a, 0x4a, 0x7a, 0x6a, 0x1a, 0x0a, 0x3a, 0x2a, 0xdb, 0xcb,
0xfb,
        0xeb, 0x9b, 0x8b, 0xbb, 0xab, 0x6c, 0x7c, 0x4c, 0x5c, 0x2c, 0x3c, 0x0c,
0x1c,
        0xed, 0xfd, 0xcd, 0xdd, 0xad, 0xbd, 0x8d, 0x9d, 0x7e, 0x6e, 0x5e, 0x4e,
0x3e,
        0x2e, 0x1e, 0x0e, 0xff, 0xef, 0xdf, 0xcf, 0xbf, 0xaf, 0x9f, 0x8f, 0x91,
0x81,
        0xb1, 0xa1, 0xd1, 0xc1, 0xf1, 0xe1, 0x10, 0x00, 0x30, 0x20, 0x50, 0x40,
0x70,
        0x60, 0x83, 0x93, 0xa3, 0xb3, 0xc3, 0xd3, 0xe3, 0xf3, 0x02, 0x12, 0x22,
0x32,
        0x42, 0x52, 0x62, 0x72, 0xb5, 0xa5, 0x95, 0x85, 0xf5, 0xe5, 0xd5, 0xc5,
0x34,
        0x24, 0x14, 0x04, 0x74, 0x64, 0x54, 0x44, 0xa7, 0xb7, 0x87, 0x97, 0xe7,
```

```
0xf7,
        0xc7, 0xd7, 0x26, 0x36, 0x06, 0x16, 0x66, 0x76, 0x46, 0x56, 0xd9, 0xc9,
0xf9,
        0xe9, 0x99, 0x89, 0xb9, 0xa9, 0x58, 0x48, 0x78, 0x68, 0x18, 0x08, 0x38,
0x28,
        0xcb, 0xdb, 0xeb, 0xfb, 0x8b, 0x9b, 0xab, 0xbb, 0x4a, 0x5a, 0x6a, 0x7a,
0x0a,
        0x1a, 0x2a, 0x3a, 0xfd, 0xed, 0xdd, 0xcd, 0xbd, 0xad, 0x9d, 0x8d, 0x7c,
0x6c,
        0x5c, 0x4c, 0x3c, 0x2c, 0x1c, 0x0c, 0xef, 0xff, 0xcf, 0xdf, 0xaf, 0xbf,
0x8f,
        0x9f, 0x6e, 0x7e, 0x4e, 0x5e, 0x2e, 0x3e, 0x0e, 0x1e
};

unsigned short int UtlComputeCRC16(unsigned char* _data_pb,const
unsigned short int& _numBytes_t);

/* End CalcCRC.hpp ============================================ */

/* Begin CalcCRC.cpp ========================================== */

#include "CalcCRC.hpp"

//============================================================================================
==================//

unsigned short int UtlComputeCRC16(
        unsigned char* _data_pb,
        const unsigned short int& _numBytes_t
        )
{
        unsigned short int k_t;
        unsigned short int highParity_t = 255;
        unsigned short int lowParity_t = 255;

   for (unsigned short int index_t = 0; index_t < _numBytes_t;
index_t++)
        {
           k_t = (unsigned short int)(*_data_pb++ ^ highParity_t);

           highParity_t = (unsigned short int)(lowParity_t ^ highCRC_ab[k_t]);

           lowParity_t = lowCRC_ab[k_t];
        }

        return(((unsigned short int)(highParity_t << 8 | lowParity_t)));

}
//============================================================================================
==================//
/* End CalcCRC.cpp ============================================ */
```

Appendix B:
Data detection/correction/encoding within science computer

Appendix C:
Flight computer I/O resources available (unused)

1 RS-232 on CPU board (max data rate 115kb/s)

1-2 RS-422/485 interfaces (max data rate 460kb/s)

About 30 unused digital I/O lines

4 10-bit ADCs (0-5VDC, few Hz sampling, rate negotiable)

2 8-bit DACs (0-5VDC, latching, updated at few Hz)

2 10-bit timers/scalars

1 PPS (GPS-derived)

10 MHz clock (GPS-derived)

5VDC 400mA (details TBD)

12VDC 100mA (details TBD)

Parallel printer port (if there's a good use...)

USB ports

3 analog video inputs

SCSI (-1 or -2 available)

Ethernet (would be shared with main data flow...)

Appendix D: Sample Commands

COMMAND TYPE	MNEMONIC	BOX DESTINATION	DATA RANGE (HEX)	SPECIAL	NOTES
BIAS ON	B	HPD	0-1	NO	0=OFF 1=ON
CALIBRATION (TEST)	C	SPARSIFICATION	0-1	NO	0=OFF 1=ON
HV ON	H	HPD	0-1	NO	0=OFF 1=ON
HV **L**EVEL	L	HPD	0-1FFE	NO	12 BIT D/A - Format = bbbbbbbbbbbb0
TRIGGER **M**ASK	M	HPD	0-FFFFFFFF	YES	32 BIT MASK + POLARITY BIT
TRIGGER **T**HRESHOLD	T	HPD	0-FFF	NO	12 BIT D/A
VA CHANNEL	V	SPARSIFICATION	0-1F	YES	1 BIT INTO 32 BIT SHIFT REGISTER
SPARSIFICATION ON	S	SPARSIFICATION	0-1	NO	0=OFF 1=ON
HOLD **D**ELAY	D	SRARSIFICATION	0-11FF	NO	8 BIT DIGITAL POT - Format = 11DD
PULSE AMPLITUDE	P	PULSE CALIBBRATION	0-FF	NO	0=OFF
CALIBRATION R**E**LAY ON	E	HPD	0-1	NO	0=-OFF 1=ON
C**O**MMAND TRIGGER	O	TRIGGER	0-1	YES	TOGGLE ON THEN OFF
RESET BOX	R	ANY	N/A	YES	TOGGLE RESET LINE LOW THEN HIGH
GLOBAL RESET	G	ALL	N/A	YES	TOGGLE ALL RESET LINES LOW/HIGH

COMMAND FORMAT *MBBAADDDDDDDD

D = HEX DIGIT
b = BINARY BIT

*	START CHARACTER
M	MNEMONIC
BB	BOX ADDRESS (2 HEX DIGITS)
AA	DEVICE ADDRESS (2 HEX DIGITS)
DDDDDDDD	DATA (8 HEX DIGITS)

INTERNAL COMMANDS

DEB**U**G (ECHO) ON	U
DEB**U**G (ECHO) OFF	u
REVISION DATE	r

Appendix E:
CREAM calorimeter readout

The CREAM Calorimeter Readout System is designed to digitize the signals from the charge sensitive amplifier ASICs, remove any data that is not significant (sparsification), and present the remaining data to the flight computer for storage and/or telemetry. The main components of this system are the ASIC boards, the sparsification modules, and the PC/104 (computer) interfaces. There is one ASIC board per HPD in both the calorimeter section and the hodoscope section. There is one sparsification module and one PC/104 interface for each of the calorimeter section and the hodoscope section. The ASIC boards are likely to be identical for both sections. The sparsification boards will differ only in quantity and size of connectors and total channels to be digitized. The PC/104 boards will be identical in both sections.

The charge sensitive amplifier ASICs are located on the ASIC boards, which attach directly to the HPD boards. The A/D converters are located on these boards, close to the ASICs to minimize noise. Serial A/D converters were chosen to minimize cabling requirements. The digital interface between the ASIC boards and the sparsification modules consists of RS-422 levels on twisted pair ribbon cables, using CMOS devices.

The sparsification boards are located close to the CREAM flight computer. These boards generate all of the timing signals required to operate the analog output multiplexers within the ASICs, and the serial A/D converters. The serial data from the A/D converters is temporarily stored in serial-in / parallel-out shift registers (implemented in Actel FPGAs). All A/D converters controlled by a sparsification board are operated simultaneously. At the end of each conversion cycle, the sparsification process transfers all significant data to dual-port memories to be presented to the CREAM flight computer. The dual-port memories are configured as ping-pong buffers, so that an event can be processed while the computer is still reading the previous event. Each HPD pixel has a unique address as determined by the ASIC channel number and the A/D number. The sparsification process consists of sequencing through each pixel channel immediately after the A/D conversion, and comparing the digital value stored in the shift register with a corresponding sparsification threshold for that channel. The comparison is performed with a simple magnitude comparator. If the data exceeds the threshold, then both the data and the corresponding address are written to the dual port memory, and the memory address counter is incremented, otherwise nothing is written to memory, and the next channel is evaluated. The sparsification threshold values are also stored in dual-port memory, so the flight computer can update them as often as required. At the end of the cycle, the total number of channels written to memory is stored in a 16-bit latch. This number, along with a 32-bit event counter number is written to the top of the memory buffer, and the memory buffers are swapped so the new buffer becomes available to the computer. There is one I/O port that is used to indicate to the computer when new data is available, and for the computer to acknowledge when it is finished reading the data.

The PC/104 board contains the address decoding circuitry for the I/O port and the block of memory. The dual port memory, which contains the sparsification threshold data, is also located on this board. The memory sections are physically split between the two boards in this manner such that only data is presented across the interface, without any fast signals or edge dependent signals at the interface. The total resources required for the PC/104 board is a block of 32 Kbytes of memory and one I/O port. Power (+5 volts) from the computer power system is also used to power the sparsification boards.

Appendix F:
Calorimeter PC/104 readout

The CREAM Calorimeter PC/104 Interface is designed to provide the interface between a Sparsification Box and the CREAM Flight Computer. The interface resides in the Flight Computer and interfaces to the computer through the PC/104 bus as a block of memory and I/O. The interface will appear to the computer as a 32 Kbytes block of memory in the address range of 0C0000H-0DFFFFH, and a 4 byte I/O address in the range of 0340H-0377H. There is no interrupt capability on the interface card. The interface to the Sparsification Box consists of 2 ribbon cable connectors of 34 pins each. Power is provided to the Sparsification Box through these connectors.

The lowest 16K block of memory is physically located external to the board (in the Sparsification Box), and is read-only. The next 8K block of memory is the sparsification threshold memory, and is physically a dual-port memory, configured as a ping-pong buffer, with the most significant address bit used to switch the buffers via a flip-flop that is toggled by a bit in the output port. The most significant address bit is not decoded for this memory, so that this memory would actually also appear at the uppermost 8K block of memory.

There is one I/O port, which only utilizes the lowest 4 bits. The hardware design is simplified to the extent that address decoding for I/O will decode a 4-byte block, of which any of the 4 addresses will be the single I/O port. The I/O bits are defined in the table below. All output functions are performed by toggling the bit high and then low.

Bit	Input Signal	Output Signal	Comments
0	MEMRDACT	MEMRDDONE	Data Available / Data Read Done
1	MEMRDERR	MEMRDERRCLR	Timeout Error / Clear Timeout Error
2	SPARSESWAP	SPARSESWAP	Swap Buffers – Sparsification Memory
3	SPARE	SPARE	

Appendix G:
Calorimeter data format

The cream calorimeter data format consists of a 128-byte header followed by the data.

```
Data Header:

Description              Data Type
Run Type                 U8    (1 = Data, 2. = Pedestal, 3. = Test pulse)
Channels                 U16
Run #                    U32
Year                     U16
Month                    U16
Day of Month             U16
Hour                     U16
Minute                   U16
Second                   U16
Day of week              U16
Day of Year              U16
DST                      U16   Daylight saving time flag
Beam Particle            U8    1(electron), 2(Proton), 3(pion+), 4(pion-)
Momentum                 SGL
Trigger Rate             SGL
Table position X         SGL
Table position Y         SGL
High Voltage[20]         20 SGL      One per box
Bias                     U32         On/Off one bit per box
Sparsification on/off    U8
Pad Bytes[41]            41 U8       41 bytes set to FF

Data:
      Sparsification On
-----------------------First Event-------------------------------------
#Channels (n)     *      U16   Channels above threshold in this event.
Unused                   U16
Event number             U32
Data(1)                  U16
Channel(1)               U16
Data(2)                  U16
Channel(3)               U16
    .
    .
    .
    .
Data(n)                  U16
Channel(n)               U16


-----------------------Second Event------------------------------------------
#Channels (n)            U16   Channels above threshold in this event.
Unused                   U16
Event number             U32
Data(1)                  U16
Channel(1)               U16
Data(2)                  U16
Channel(2)               U16
    .
    .
    .
    .
Data(n)                  U16
Channel(n)               U16
```

```
------------------------------------------------------------------------------
      |
      |
      |
      |
-----------------------Last Event-----------------------------------------
#Channels (n)           U16   Channels above threshold in this event.
Unused                  U16
Event number            U32
Data(1)                 U16
Channel(1)              U16
Data(2)                 U16
Channel(2)              U16
    .
    .
    .
    .
Data(n)                 U16
Channel(n)              U16
-----------------------Run end header-----------------------------------
End of data (FFFFFFFF)  U32
Year                    U16
Month                   U16
Day of Month            U16
Hour                    U16
Minute                  U16
Second                  U16
Day of week             U16
Day of Year             U16
DST                     U16   Daylight saving time flag
Status                  U8
```

Appendix H:
Calorimeter command system format

The CREAM Command System is designed to accommodate the commanding requirements of the CREAM Calorimeter. It can also provide digital serial commands to other subsystems, as required. Commands are passed from the CREAM Flight Computer to the Command System by a standard RS-232 serial port using ASCII characters. The Command System formats the commands as appropriate, and sends them to the electronics boxes as RS-422 differential signals, consisting of Data, Clock, Address Enable, and Data Enable. A reset signal is included in the command interface, which can be activated as a global reset, or on a per box basis.

The processor chosen for the Command System is an Atmel 89S8252. This processor is based on the Intel 8051 architecture, which provides a reasonable instruction set, including good bit-level instructions. There are a wide variety of development tools available for this architecture. This device has the additional advantage of internal flash memory, which is in-circuit programmable through a simple serial interface. This quick re-programmability feature greatly reduces development and debugging effort. Redundant program memory will be used to increase reliability, by including provision to switch the execution memory between internal flash and external EPROM. This particular device is readily available in industrial temperature range in the 44 pin PLCC package. The device can operate from 0-33 MHz, and we will likely use 3.684 MHz to keep the power very low, while still maintaining serial port speeds up to 19,200 Baud.

The command interface to the electronics boxes consists of five signals output with RS-422 differential line drivers on a 14-pin ribbon cable connector to be used with twisted pair ribbon cable. The line drivers have 33-ohm series resistors on the outputs to terminate the reflected signal and minimize radiated noise. There is a separate connector provided for each box to receive commands. The connector is configured as shown in the following table.

SIGNAL	NAME	PIN (+)	PIN (-)
Address Enable	ADDREN	1	2
Data Enable	DATAEN	3	4
Data	DATA	5	6
Clock	CLOCK	7	8
Reset	RESET	9	10
Spare - NC		11	12
Ground	GROUND	13	14

The commands sent to the Command Processor via the serial port consist of strings of ASCII characters. Each command begins with a start character (*), followed by 12 hex characters, representing the 6-byte command. The command format includes 1 byte for box address, 1 byte for device address, and 4 bytes of data. The box address is used by the Command Processor to determine which box connector should receive active address and data enable lines. The device address plus 4 data bytes are transferred to the appropriate box. The timing diagram and command lists are maintained in separate documents.

CREAM Flight Software Flowchart